A New Topological Landscape Algorithm for Visualization of Scalar-Valued Functions

William Harvey Yusu Wang

October 2, 2009

Abstract

Visual representation techniques enable perception and exploration of scientific data. Following the topological landscapes metaphor proposed in [1], we provide a new algorithm for visualizing scalar functions defined on manifolds of arbitrary dimension. For a given input function our algorithm produces a collection of two-dimensional terrain models whose critical points and topological feature persistence are identical to those of the input function. The algorithm exactly preserves the volume ratios of corresponding topological features. We also introduce an efficiently computable metric on terrain models that is useful in exploring the space of possible terrain models for a given function and can be used to augment the data visualization pipeline in other ways.

1 Introduction

Visualization and analysis of scalar functions $f : \mathbb{D} \to \mathbb{R}$ for various domains $\mathbb{D}$ constitutes a significant facet of the scientific process. Scalar functions are prevalent in a variety of disciplines and occasionally admit simple and natural visualizations when $\mathbb{D}$ takes certain forms. For example, grayscale digital images with dimensions $(w \times h)$ are scalar functions with domain $\mathbb{D} = \{(x, y) \in \mathbb{N}^2 \mid 0 \leq x < w, 0 \leq y < h\} \subset R^2$ and range $\mathbb{R}$ and are directly and easily visualized via rendering to a raster display.

As $\mathbb{D}$ assumes different forms, visualizing its scalar functions can become more difficult. Moving from the world of easily-visualized 2D digital images to 3D volumetric data poses new challenges, including possible occlusion of regions of the data under analysis, and additional data processing and storage requirements. As the number of dimensions of $\mathbb{D}$ becomes large, an effective visualization technique may no longer be obvious.

Various dimensionality reduction techniques including principal component analysis (PCA) [13], ISOMAP [24], Laplacian eigenmaps [4], locally-linear embedding (LLE) [20] and others are often useful for scalar function visualization, especially when the function is available as samples in the form of point cloud data. There is significant variety of dimensionality reduction techniques to choose from, providing a degree of flexibility in terms of which geometric properties of the original data will be preserved; cf. [26]. However, there is no guarantee that dimensionality reduction will preserve the topological structure of f. This shortcoming can limit their usefulness in understanding the structure of scalar functions on some domains.

Fortunately, a variety of computational approaches for topological analysis of scalar functions have been cultivated. The contour tree [5] is one such object, and has found applications in geographic information systems [22, 10], volume visualization and segmentation [2, 9, 6], among other areas. Figure 1 displays a simple example of a contour tree for a scalar function defined on a two-dimensional domain. An $O(n \log n)$ algorithm for computing the contour tree on two-dimensional manifolds (but $O(n^2)$ for higher dimensions) was presented by de Berg and Van Kreveld [27]. Here, n is the number of vertices in the input mesh. Tarasov and Vyalyi [23] improved the algorithm of [27] to yield a time complexity of $O(n \log n)$ for three dimensional domains. Carr et al. [7] presented an $O(n)$ algorithm that is simple to implement and works on input meshes of arbitrary dimensions. Recently Shigeo et al. [21] proposed a method for approximating the structure of the contour tree by applying a novel metric to point cloud data and then performing standard dimensionality reduction to yield a new point cloud whose structure approximates the structure of the contour tree.

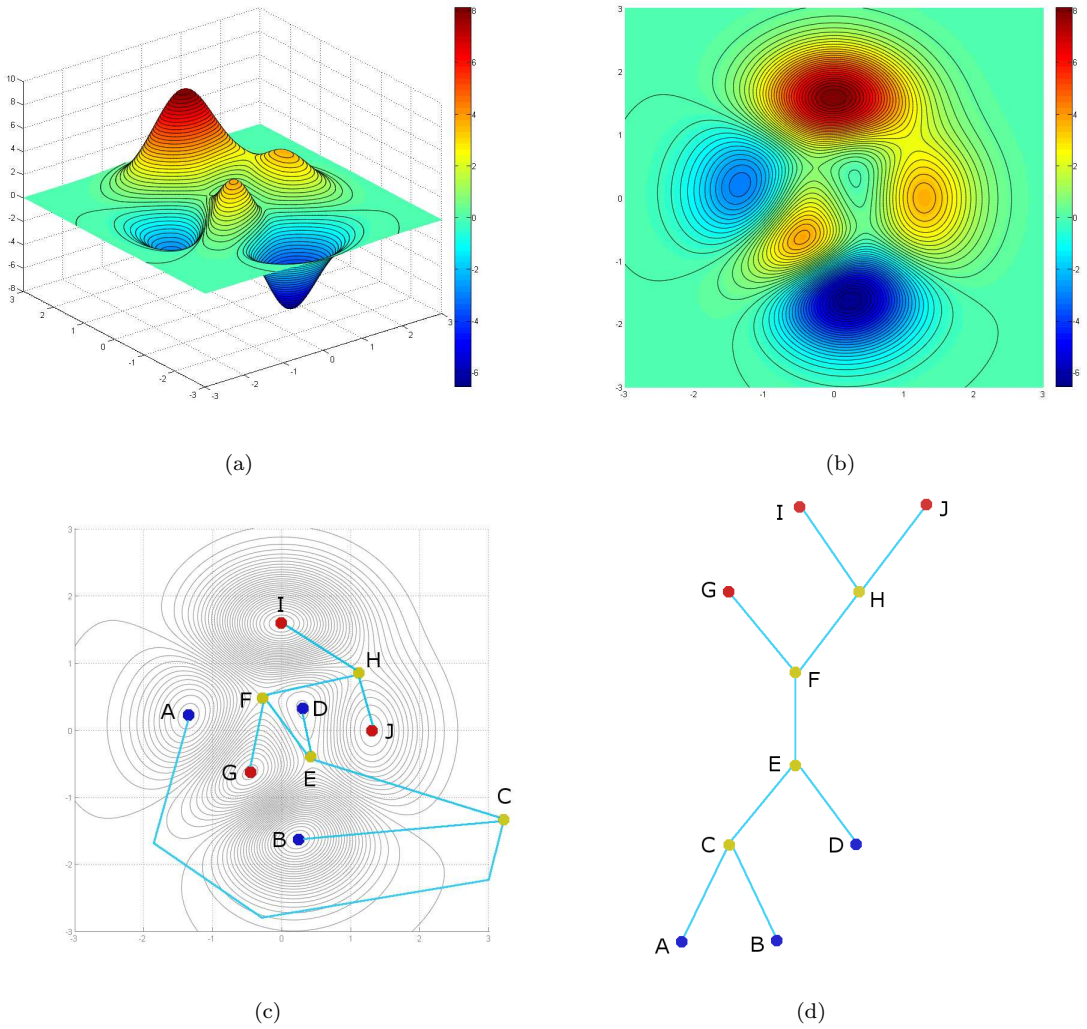


Figure 1: A simple contour tree example. A detailed exposition is reserved for section 3. (a) A simple function defined on a subset of the plane. The black curves indicate a few of the contours of the function. (b) Nadir image of the function. From this perspective the structure and nesting relationships of the contours are clearly visible. (c) The critical points of the function; red indicates a local maximum, blue a local minimum, and yellow a saddle point. (d) The contour tree of the function, derived by contracting each contour to a single point.

Another powerful and widely used visualization technique is the treemap [14]. Treemaps are useful for visualizing hierarchically-organized datasets by recursively subdividing a subset of $\mathbb{R}^2$ such that the area of each subset corresponds to a given attribute of the dataset. These techniques have been applied to a variety of problem domains, cf. [3, 25, 28]. The basic treemap construction algorithm (dubbed slice-and-dice) uses horizontal and vertical lines to partition the plane into a collection of rectangular regions. Other methods have been proposed for computing the treemap layout including voronoi treemaps [3] and a method [18] using convex polygons with provable bounds on the aspect ratios of subregions. While treemap algorithms have not to our knowledge been used explicitly for visualization of scalar fields, they are an integral component in the approach proposed in this paper.

While we adopt the topological landscapes metaphor proposed in [1], our work addresses several shortcomings. First, unlike the algorithm in [1], our method exactly preserves the volume ratios of the topological components the input function f. Second, our algorithm does not use nonlinear optimization techniques and has a guaranteed time complexity of $O(nk^2)$, where n is the number of critical points of f and k is an integer parameter controlling complexity of the output mesh. Third, our method is general in that different treemap-style algorithms can be chosen to achieve desired aesthetic or geometric properties of the resulting visualization.

Finally, in contrast to [1], our method is not limited to generating a single visualization of f. Rather, we combine dimensionality reduction techniques with a novel metric on the space of the possible visualizations of f to enable realtime interactive exploration of the possible views of f. This additional mechanism is a powerful and useful augmentation to the visualization process. We conduct several experiments to illustrate the effectiveness of our terrain generation and exploration method on both low-dimensional ($\mathbb{R}^3$) and high-dimensional ($\mathbb{R}^{>1000}$) datasets.

2 Related Work

Since this work is directly inspired by the topological landscapes algorithm of [1], we summarize it in this section.

The topological landscapes algorithm takes as input a contour tree T of a function f, and produces a meshed terrain model M where the height of each vertex reflects its corresponding function value of f. M is constructed such that it has identical critical point function values and persistence of topological structures as f. The first step is to compute a *branch decomposition* of T, which groups the edges of T such that each group is a path through T with monotone function value, and the union of all branches is T. When a root branch is selected in the contour tree, the branch decomposition takes on a hierarchical structure (and is constructed recursively), where the child branches C of a given branch b are those non-parent branches that are incident to b.

The next step is to recursively construct a terrain mesh (a SOAR hierarchy [16]) using the structure of the branch decomposition as a guide. Each branch in the branch decomposition is mapped to a square region on the terrain mesh, and the mesh within this square region is refined to ensure that there is room to insert the child branches with enough space between them to preserve the topological structure of f. This process continues until every branch in the branch decomposition has been mapped to a square region on the terrain mesh.

An optional contour tree rebalancing algorithm can be used to reduce the depth of the branch decomposition hierarchy and reduce the resulting terrain complexity by making its topological structure more obvious. However, performing this rebalancing operation results in a branch decomposition that no longer exactly reflects the nesting relationships of the contour lines of f.

Once the terrain mesh has been constructed, the areas of the square regions are adjusted using an iterative optimization procedure. The goal of this optimization is to match the area of each square region with the area of its corresponding branch in the branch decomposition. [1] reports that the maximum error of a single branch encountered in all attempted experiments was 8.8%, and the largest average error for a dataset was 1.4%. The authors report that the runtime of this optimization procedure is parameter and data dependent, and with inappropriate parameters may not converge.

3 Preliminaries

In this section we formally introduce the contour tree and its relationship to a scalar function defined on a manifold. For a detailed exposition of the theoretical underpinnings we refer the reader to [17].

Definition Let $f : \mathbb{M} \to \mathbb{R}$ be a scalar function defined on a manifold $\mathbb{M}$. An *isosurface* of f is a subset of the domain for which f holds some constant value called an *isovalue*. More specifically, given an isosurface $S_{f(x)} \subset \mathbb{M}$ corresponding to an isovalue $f(x)$, $S_{f(x)} = f^{-1}(x)$. A *contour* is a single connected component of a given isosurface.

Definition The *contour tree* T of f corresponds to a continuous contraction of each possible contour of f to a single point [19], i.e. it is the quotient space of points belonging to the same contour of f.

There is a 1-to-1 correspondence between the points of T and the contours of f. The leaves of T correspond to local extrema (contours of f with genus zero), with lower leaves of T corresponding to the local minima of f and upper leaves of T corresponding to the local maxima of f. Points of T with degree two such as those found along the branches of T correspond to contours of f with genus one. Finally, points of T with degrees greater than two correspond to contours of f with genera greater than one whose single point of self-intersection corresponds to a saddle point of f.

Definition Let $\tau_f : \mathbb{M} \to T$ be the map taking each point in the domain of f to its corresponding point in the contour tree. Let x be a critical point of f. Let (U, V) be a partition of T at point x, i.e. $U \cup V = T, U \cap V = \emptyset, x \in U$, with U connected. The *topological feature* corresponding to U is defined as $\Phi_U = \tau_f^{-1}(U)$. The *topological persistence of* Φ_U is $p(\Phi_U) = \varnothing(f(\Phi_U)$, where $\varnothing$ denotes the diameter of an interval in $\mathbb{R}$.

Thus T captures all critical points of f as well as the persistence of topological features formed by its contours. Due to this fact, the contour tree data structure is a useful tool for scalar field visualization.

One disadvantage of the contour tree as a visualization tool is that it is an abstract structure without a natural embedding and it can be difficult to interpret in its entirety, especially when it is very large or complex. A common approach to dealing with this problem include symbolic alteration of f by directly modifying the structure of T to achieve a desired degree of simplification [8], or balancing the contour tree [1].

4 Algorithm

The proposed algorithm takes as input a contour tree T of a function $f : \mathbb{M} \to \mathbb{R}$ defined on a manifold $\mathbb{M}$ and produces a function $g : (I = [0, 1])^2 \to \mathbb{R}$ defined on the unit square such that the contour tree of g is precisely T. Additionally, the areas of the topological components of g are exactly proportional to the volumes of the topological components of f. Throughout this paper, the term 'terrain model' is used to refer to the function g.

First, it may be necessary to apply topological perturbation to f in the form of small structural changes to T to ensure that the sum of the indices of the critical points of f is 2; i.e. it must be possible to embed f in $\mathbb{S}^2$ without introducing any degenerate critical points.

Next, a single local extremum x_∞ of f must be chosen to map to the boundary $\partial(I^2)$ under an implicit sphere projection $\pi : \mathbb{S}^2 \to I^2$. Different choices of x_∞ will yield different layouts of the topological structures, which allows for some flexibility in which terrain model to choose for visualizing f. For example, one could choose the terrain model with the lowest average aspect ratio of topological components or the model which maps a particular topological feature to the perimeter of I^2.

To assist in choosing x_∞, we propose an efficiently computable metric on terrain models that can be used to interactively explore the space of possible terrain models via dimensionality reduction. This metric can also be used to cluster and summarize the possible terrain models or to perform various kinds of geometric analysis or processing.

Once x_∞ has been chosen, the contours of f are embedded in I^2 using an efficient recursive algorithm. We then extend the function values of the embedded contours to the rest of I^2 by meshing the interior region bounded by adjacent nested contours and interpolating g in a manner that facilitates visual interpretation of the resulting terrain model.

4.1 Contour Tree Perturbation and Simplification

Since the algorithm relies on the assumption that of f can be embedded in $\mathbb{S}^2$ without introducing degenerate critical points, we may need to perturb f (via operations on T) to ensure that the sum of the indices of the critical points of f is $\chi(\mathbb{S}^2) = 2$. Each saddle node s_i in T with a degree $deg(s_i)$ greater than two is topologically perturbed by transferring its incident edges to nearby noncritical nodes in T_f. The perturbation of s_i ceases when $deg(s_i) = 3$. This process simulates breaking apart clusters of coincident saddle points of f until all saddle points are isolated.

At the end of the perturbation process, the sum of the indices of the critical points is 2, all nodes in the contour tree have degrees 1 (leaf nodes) or 3 (saddle nodes), and f can be embedded in $\mathbb{S}^2$ without introducing degenerate critical points.

Optionally, the contour tree can be simplified by eliminating branches corresponding to topological features whose persistence is below a user-specified threshold. This simplification process is used in [1] to reduce the visualization clutter of highly complex contour trees.

4.2 Embedding Contours in I^2

To embed the contours of f in I^2, we take advantage of the idea that f can be embedded in $\mathbb{S}^2$, and a sphere projection can then be used to embed the contours in I^2. To begin, it is worthwhile to investigate the relationship between the inclusion pattern of the embedded contours and the structure of the contour tree.

Recall that $\pi(x_\infty) = \partial(I^2)$. Thus, $\pi(x_\infty)$ is the outer-most embedded contour, and contains all other contours in its interior. The contours along any edge of T form nested rings on I^2. A contour corresponding to a saddle node of T can assume one of two possible configurations depending on the relative position of x_∞. Figure 3 illustrates these two possible configurations.

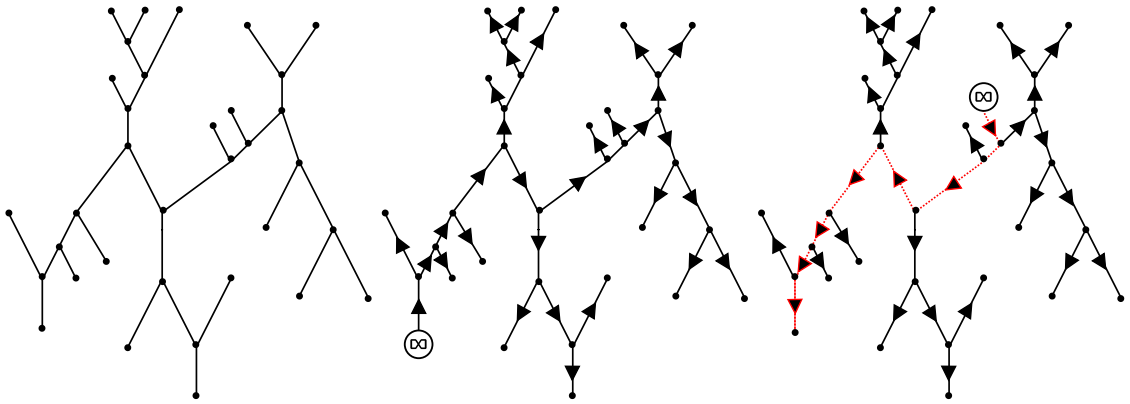
The inclusion relationships of all contours of f can be summarized by imposing edge directions on T to yield an *inclusion arborescence* of T. Under this representation, a directed path from node x_i to x_j indicates that $\pi(x_i)$ contains $\pi(x_j)$. Choosing a different x_∞ will yield a different inclusion arborescence, as illustrated in figure 2.

Although there are two configurations of saddle node contours (see figure 3) the algorithm embeds both types of contours identically. Figure 4 illustrates how contours corresponding to type I and II bifurcations are embedded in I^2 when using a simple slice-and-dice approach. The rectangular contours of figure 4 can assume other shapes by employing a different treemap algorithm.

The algorithm for embedding the contours of f in I^2 begins by mapping x_∞ to the square boundary of I^2. The contours are then embedded in a recursive fashion corresponding to a depth-first traversal of T. After processing node x_i of T and creating embedded contour r_i, there are two possibilities: (1) If the next node x_j encountered during recursion is a saddle node, then r_j is generated by scaling r_i such that the area of the region bounded by r_i and r_j is in proportion to the volume of the topological structure corresponding to edge e_{ij} of T. At this point, r_j is then split into two sub-regions r_{j0} and r_{j1} with areas proportional to the topological structures that they bound. Recursive processing of r_{j0} and r_{j1} ensues. (2) If the next node x_j encountered during recursion is a leaf node, then recursion terminates.

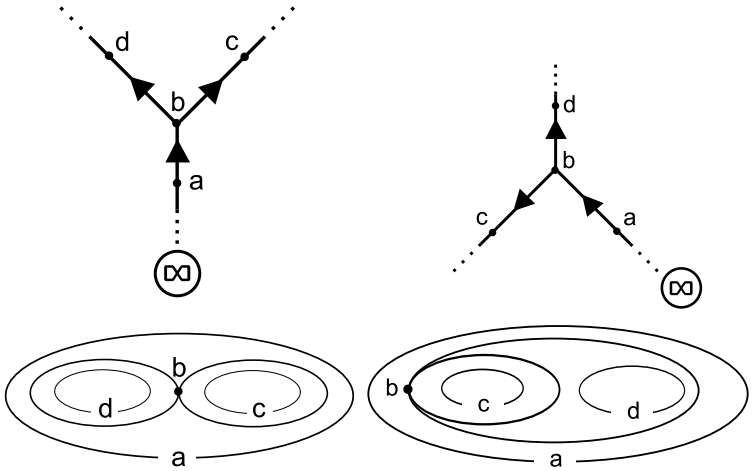
4.3 Choice of x_∞

The choice of x_∞ can impact the overall appearance of the generated terrain model, and should be guided according to the application. It may be known a priori which local extremum of f would be the ideal x_∞, e.g. when there is some topological feature of f that is well suited as a boundary for the terrain model. However, such domain knowledge is not always readily available, especially when analyzing unfamiliar data.



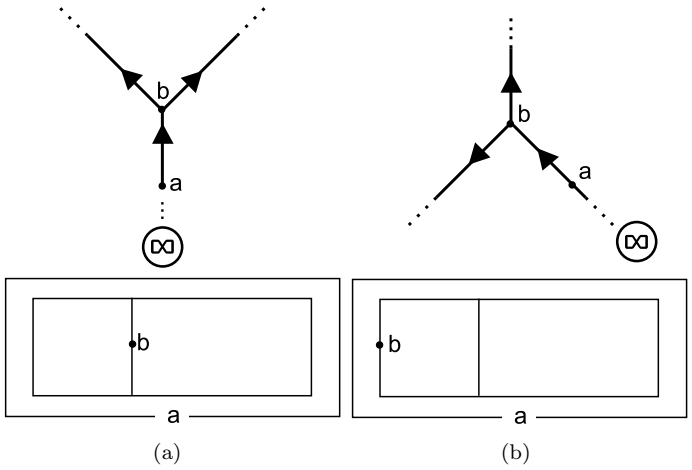
(a) (b) (c)

Figure 2: (a) A contour tree T for a function f. Upper and lower leaves correspond to local maxima and minima of f, respectively; nodes with degree 3 correspond to saddle points of f. (b) Arborescence of T when a local minimum is selected as x_∞. The edge directions indicate the inclusion relationships of the contours of f when embedded in I^2. (c) Arborescence of T when a different local extremum is selected as x_∞. Note that when x_∞ changes, the inclusion relationships of the embedded contours reverse only for contours corresponding to the path connecting the old and new x_∞ as shown in red.



(a) (b)

Figure 3: The inclusion relationships of contours at saddle nodes of T for a given choice of x_∞. Edge directions indicate the inclusion relationships of the embedded contours. (a) Example of a type I contour bifurcation. (b) Example of a type II contour bifurcation.



(a) (b)

Figure 4: Scheme for embedding contours. In both cases, the region between curves a and b is proportional to the volume of the topological feature of f corresponding to edge e_{ab} of T.(a) Embedding a contour with a type I bifurcation. (b) Embedding a contour with a type II bifurcation. Note that the embeddings are identical, providing a simple algorithm for producing the embeddings.

To assist a user in exploring the space of possible terrain models of f and in making appropriate choices of x_∞, we propose an efficiently computable metric on terrain models which captures the visual difference between two models as the total weighted area of one terrain model that must be inverted to produce the other terrain model.

Let g_i be a terrain model which maps point $\mathbf{x}_i$ to infinity under the map π, and having inclusion arborescence I_i. Let g_j be another terrain model which maps point $\mathbf{x}_j$ to infinity under the map π, and having inclusion arborescence I_j. The inclusion arborescences are identical except that the path connecting $\mathbf{x}_i$ to $\mathbf{x}_j$ in I_i is reversed in I_j. When the point at infinity is changed from $\mathbf{x}_i$ to $\mathbf{x}_j$, the inclusion relationships of the contours separating $\mathbf{x}_i$ from $\mathbf{x}_j$ are reversed. The total area that must be inverted to transform g_i into g_j is the sum of the areas of the contour tree edges connecting them. Taking the product of the area of an edge and its range of function values yields a relaxed earth mover's distance which intuitively encodes the amount of soil which must be transported to convert one terrain into another.

This metric can be efficiently computed by first assigning a weight w_{ij} to each edge e_{ij} of T:

$$w_{ij} = Area(e_{ij}) \mid f(\mathbf{x}_i) - f(\mathbf{x}_j) \mid \tag{1}$$

The unique path $P(i,j) = \{e_{i\alpha_1}, e_{\alpha_1\alpha_2}, ..., e_{\alpha_m j}\}$ connecting x_i to x_j in T is calculated using breadth-first search. Then the distance between g_i and g_j can be computed as follows:

$$d(g_i, g_j) = \sum_{e_{\alpha\beta} \in P(i,j)} w_{\alpha\beta} \tag{2}$$

For our experiments, we compute $d(g_i, g_j)$ for all pairs of local extrema (x_i, x_j) and use ISOMAP [24] to represent each terrain model g_i as a 2D point $y_i \in \mathbb{R}^2$. Since the difference in appearance of two terrain models is captured by the metric d, the resulting 2D point cloud provides a visualization of the variations in appearance of the terrain models and can facilitate selection of one or more representative models from the pool of all choices. We provide a user interface in which a user may click on any point x in the 2D point cloud to instantly bring up a terrain model that uses $x = x_\infty$. This allows a user to rapidly find a visualization which accentuates any features of interest.

4.4 Mesh Generation

With x_∞ chosen, the contours of f can be embedded in I^2; the values of g along these embedded contours are now defined. The remaining values of g can be recovered by interpolation guided by the embedded contours. A simplicial mesh S is constructed on I^2 with the embedded contours preserved as a subset of the 1-simplices of S. For this discussion, we assume without loss of generality that the contours have been embedded using the slice-and-dice treemap approach.

In the first case, the region $R \subset I^2$ is a rectangular annulus with an outer boundary rectangle r_1 and inner boundary rectangle r_2. The values of g between r_1 and r_2 are linearly interpolated, so a simple triangulation of the interior of R is performed which connects each edge segment of r_1 to a point of r_2 and vice versa.

In the second case, the region R forms a rectangle. Such regions correspond to terminal edges in the contour tree, and will contain a single local extremum somewhere in their interior. In this case, we place the local extremum at the center of R and add additional vertices and edges in a rectangular k by k lattice pattern to the interior of R. The parameter k is user-specified parameter and trades visual quality for mesh simplicity. We have found that $11 \leq k \leq 21$ provides a sufficient range of visual quality while maintaining mesh economy. Note that k should be odd, as we would like R to have a single vertex at its center to represent its local extremum. The additional vertices will provide some flexibility in how the values of $g(R)$ are interpolated, as is discussed in section 4.5.

4.5 Function Interpolation

Recall that values of g within rectangular annuli are computed using simple linear interpolation of their inner and outer boundary function values.

Consider the set of vertices V of a rectangular region R corresponding to a leaf edge of T. Let $B = \partial V$ be the vertices on the boundary of V, v_c the vertex at the very center of R, and $U = V \setminus (B \cup \{v_c\})$ be all other vertices of R. The quantities $g(B)$ and $g(v_c)$ are defined, taking on the values of the critical points of f to which they correspond. However, $g(u_i), u_i \in U$ are still undefined and must be calculated via interpolation of $g(B)$ and $g(v_c)$.

This interpolation can be addressed as an instance of the discrete Dirichlet problem [12]. More specifically, we wish to determine a quantity $\alpha_{u_i} \in [0, 1]$ for each $u_i \in U$ and define $g(u_i)$ as a convex combination of $g(B)$ and $g(v_c)$:

$$g(u_i) = \alpha_{u_i} g(v_c) + (1 - \alpha_{u_i}) g(B) \tag{3}$$

We first construct the graph Laplacian matrix L for region V given by

$$L_{i,j} := \begin{cases} \sum_{i \neq k} w_{ik} & \text{if } i = j \\ -w_{ij} & \text{otherwise.} \end{cases} \tag{4}$$

The Laplacian weight w_{ij} of vertices v_i and v_j of R is formulated:

$$w_{ij} = e^{-\frac{d_{ij}^2}{4k^{3/2}}} \tag{5}$$

where d_{ij} is the distance between the rectangular lattice coordinates of v_i and v_j. The last step is to compute the alpha values for the nodes in U using the method described in [12], which amounts to solving a small system of $|U|$ linear equations. This interpolation process using the weighting scheme in equation 5 yields smooth dome-like structures that facilitate visual interpretation. We have found that by scaling down the unknown alpha values by 5% allows the local extremum within R to be clearly visible as a salient peak at the center of the dome. This peak provides an additional visual cue to facilitate visualization.

Note that this alpha mask only needs to be computed once, as all rectangular regions contain the same rectangular lattice pattern of interior vertices. With the alpha mask in hand, g is interpolated to undefined regions of R using equation 3.

5 Discussion

Given a contour tree T with n nodes, and provided x_∞ is specified, the time complexity for producing a terrain model is $O(nk^2)$, where n is the number of nodes of T and k is the parameter specifying the width and height of the vertex lattice to insert within rectangular regions corresponding to the leaf edges of T. This marks a notable difference from [1] for which a computational complexity is unknown due to reliance on a data and parameter dependent optimization technique.

Another advantage of the proposed technique is availability of the metric d for computing the distance between two terrain models. While we use this metric as a means of selecting various values of x_∞ for a dataset, there are multiple ways that it could be applied throughout the visualization pipeline.

6 Results

6.1 3D Volume Datasets

As in [1], a collection of 3D volume datasets available from http://www.volvis.org was processed using the proposed algorithm. For these experiments, we choose x_∞ to match the outer components of the terrain models of [1]. We also illustrate the usefulness of computing embeddings of the possible terrain models in terms of exploring the space of visualizations. Finally, we generate terrain models using both the slice-and-dice and voronoi treemap embeddings of the contours to demonstrate the flexibility of the proposed approach.

Figures 5, 6, 7, and 8 illustrate the terrain models generated for these datasets and compare with results from [1]. For these experiments, we choose x_∞ to match the outer components of the terrain models of [1].

One should note that minor differences may occur between the topology of terrains generated by [1] and the proposed method. We hypothesize that these differences may be a result of cancellation of insignificant topological features by [1], whereas we do not perform any sort of topological cancellation. Also, one should note the difference between using the slice-and-dice contour embedding method and the voronoi contour embedding method. Using other treemap algorithms to perform the contour embedding provides some flexibility in how the floorplan (the orthogonal projection of all contours onto the ground plane) is constructed; for example, the algorithm proposed in [18] could provide theoretical guarantees on the aspect ratios of all contours.

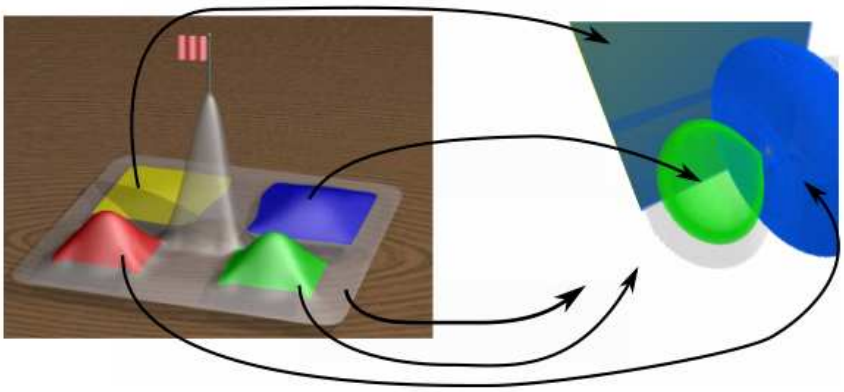
6.2 Protein Folding

One interesting application of this algorithm is for visualizing functions defined on manifolds of dimensions 4 or more. The objective of the experiments on protein folding data was to try to recover the energy landscape of the conformational space of a particular protein. A replica exchange molecular dynamics (REMD) simulation was performed to simulate a protein folding from various starting conformations under fluctuating environmental temperature, yielding a dataset of 20000 protein conformations. The backbone of each conformation consists of $n = 46$ carbon atoms which we use to compute a $\binom{n}{2} = 1035$ dimensional vector of pairwise distances describing the protein shape. For these experiments, we reduce the set of protein conformations to 2000 samples using the method of Gonzales [11].

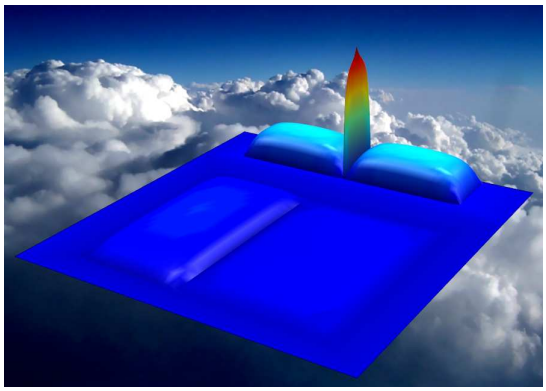
Initial experiments were conducted using traditional dimensionality reduction techniques such as PCA, ISOMAP, and Laplacian eigenmaps to try to visualize the energy landscape of the protein. However, the resulting landscapes were difficult to interpret visually as the topological structure of the original function was not well preserved using these techniques. Figure 9 illustrates an attempt at generating an energy landscape using a combination of PCA and Delaunay triangulation.

The proposed method provides a framework that facilitates interpretation of the protein folding data and helps to discover the funnel shape of the energy landscape.

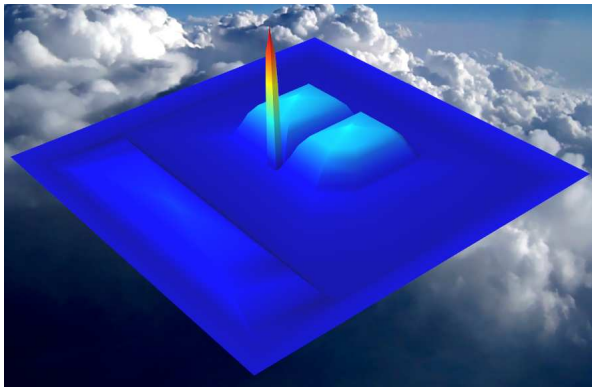
The proposed algorithm produced an terrain model that is easy to interpret and captures the hypothesized structure of the energy landscape and folding funnel. Figure 10 illustrates terrain models of the protein energy



(a)

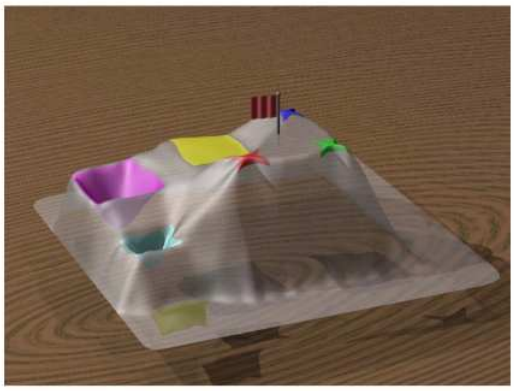


(b)

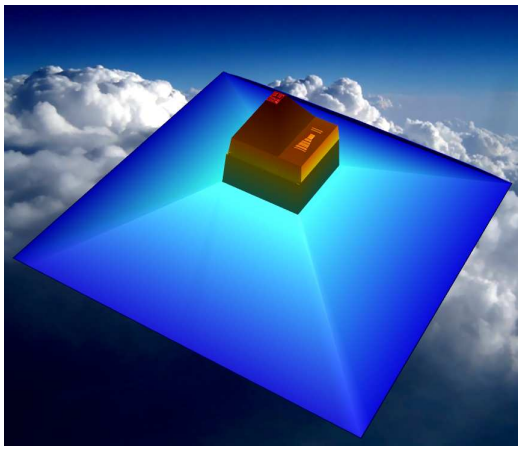


(c)

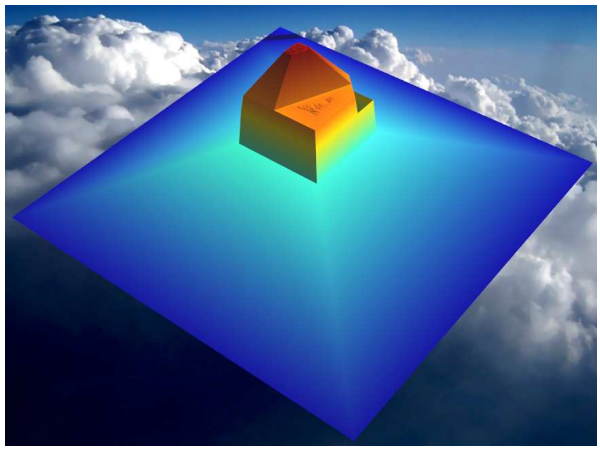
Figure 5: Hydrogen dataset. (a) Results of [1] with average volume-to-area projection error of 1.1%. (b) Results using the proposed algorithm with slice-and-dice contour embedding. (c) Results using the proposed algorithm with voronoi contour embedding.



(a)

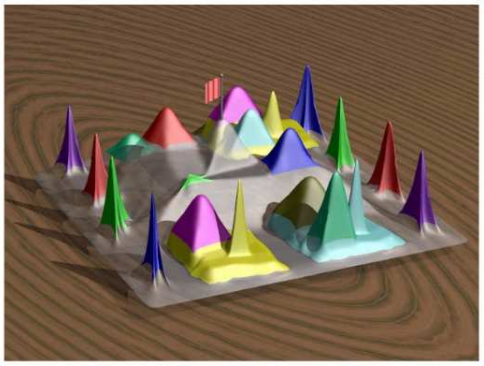


(b)

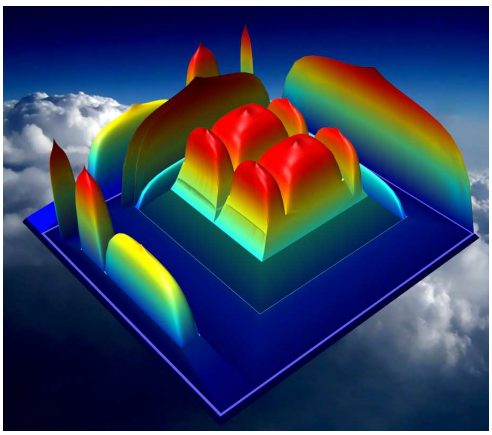


(c)

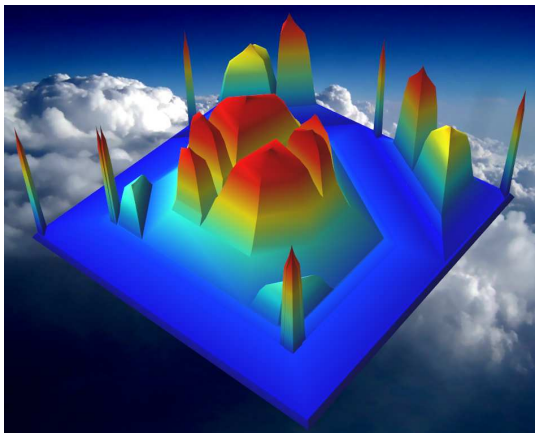
Figure 6: Nucleon dataset. (a) Results of [1] with average volume-to-area projection error of 0.6%. (b) Results using the proposed algorithm with slice-and-dice contour embedding. (c) Results using the proposed algorithm with voronoi contour embedding.



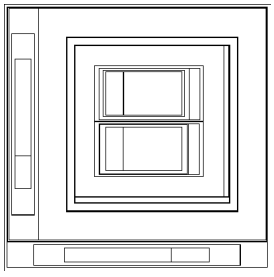
(a)



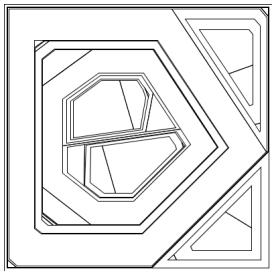
(b)



(c)



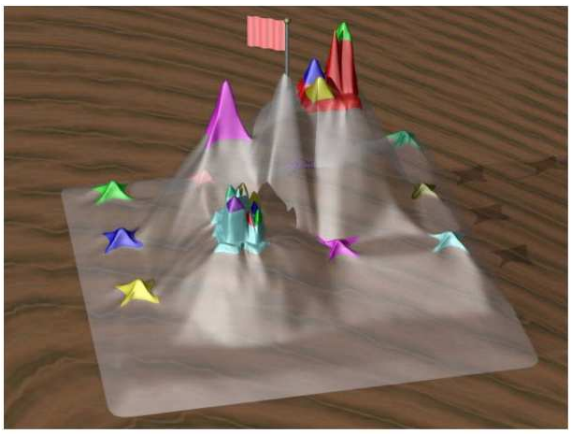
(d)



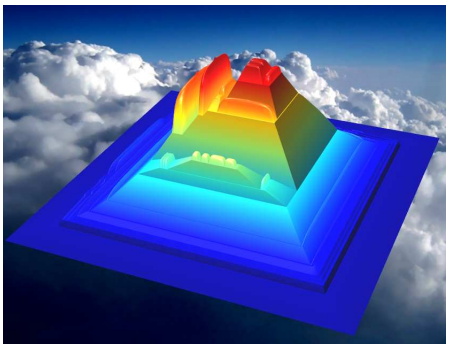
(e)

(f)

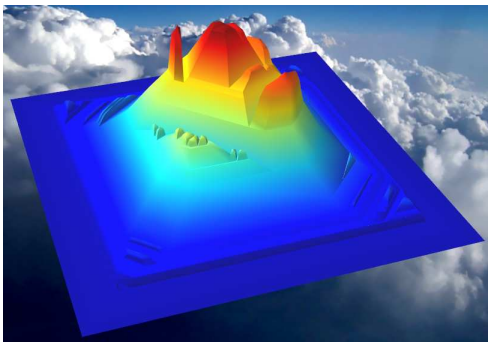
Figure 7: Neghip dataset. (a) Results of [1] with average volume-to-area projection error of 0.6%. (b) Results using the proposed algorithm with slice-and-dice contour embedding. (c) Results using the proposed algorithm with voronoi contour embedding. (d-e) Floorplans of (b-c), respectively. (f) Space of possible terrain visualizations. Each thumbnail represents a nadir view of a possible terrain model.



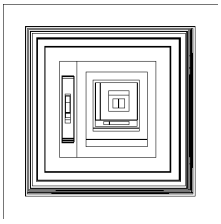
(a)



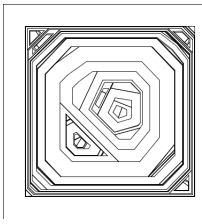
(b)



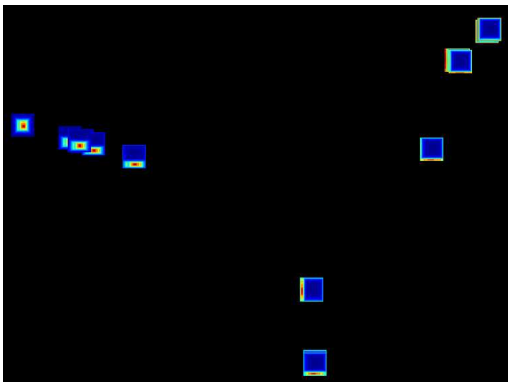
(c)



(d)



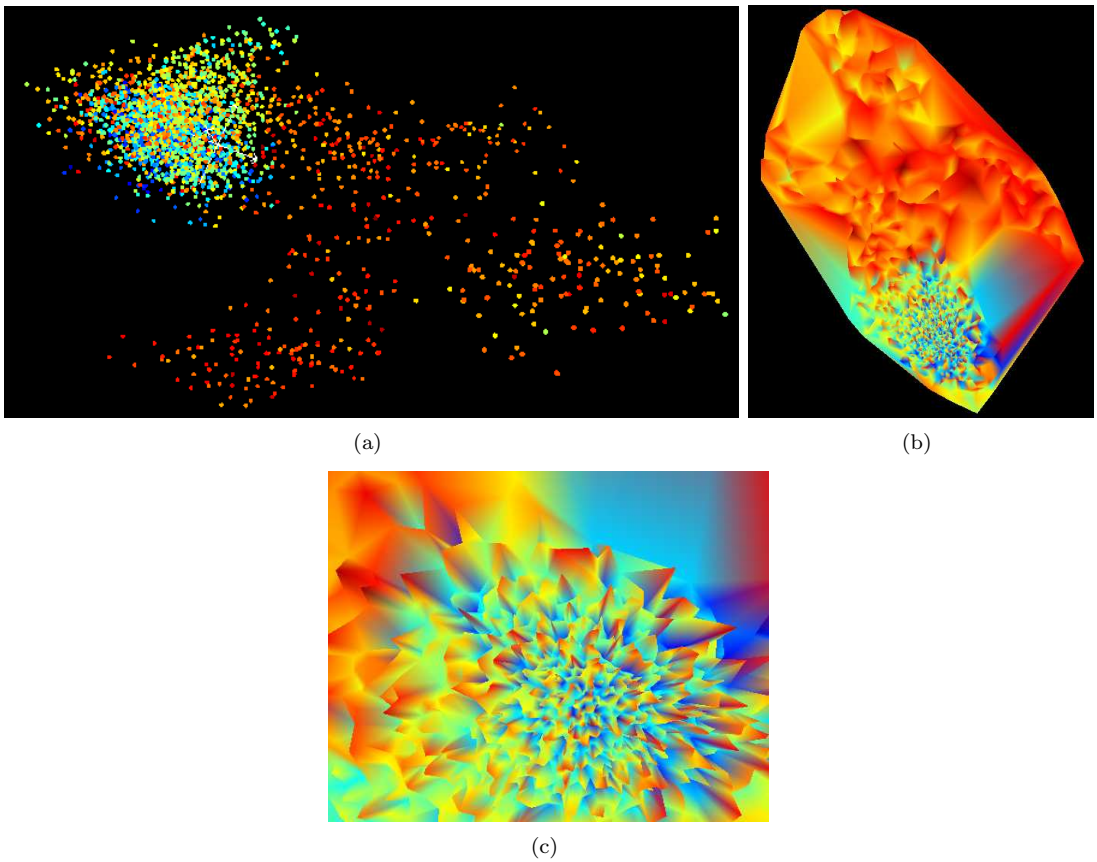
(e)



(f)

Figure 8: Fuel dataset. (a) Results of [1] with average volume-to-area projection error of 0.6%. (b) Results using the proposed algorithm with slice-and-dice contour embedding. (c) Results using the proposed algorithm with voronoi contour embedding. (d-e) Floorplans of (b-c), respectively. (f) Space of possible terrain visualizations. Each thumbnail represents a nadir view of a possible terrain model.

landscape produced using the proposed algorithm as well as the algorithm of [1]. The contour tree of the high-dimensional point cloud was computed by first approximating the (assumed) underlying manifold using a k-nearest neighbor graph followed by the contour tree algorithm of Carr et al [7].



(a) (b)

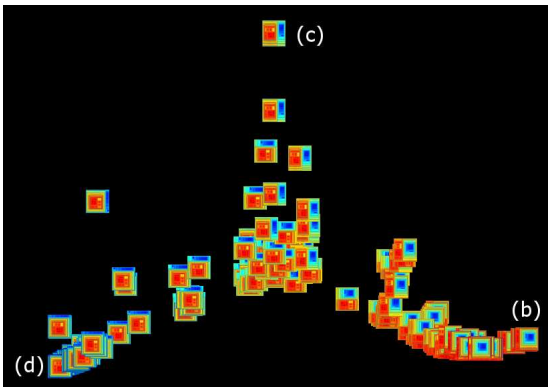
(c)

Figure 9: An attempt at producing an energy landscape visualization of the protein conformational space using PCA. (a) Conformational points reduced to 2 dimensions using PCA. (b) Terrain produced from points of figure (a) via Delaunay triangulation. (c) Additional view of the energy funnel containing the global energy minimum.

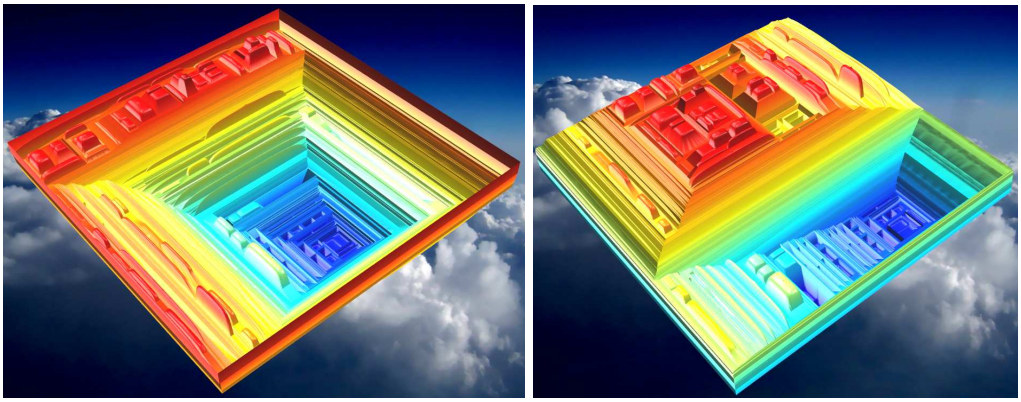
7 Conclusions and Future Work

We adopt the topological landscape paradigm proposed in [1] and propose a new algorithm for visualizing scalar-valued functions defined on manifolds of arbitrary dimension. Given a contour tree T of a function $f : \mathbb{M} \to \mathbb{R}$, our algorithm produces a new function $g : I^2 \to \mathbb{R}$ with the same critical point values and persistence of topological structures as f. Additionally, the areas of the topological structures of g exactly mimic the volumes of their counterparts in f. We propose a metric on the possible terrain models of T that facilitates the process of exploring the space of possible visualizations of f.

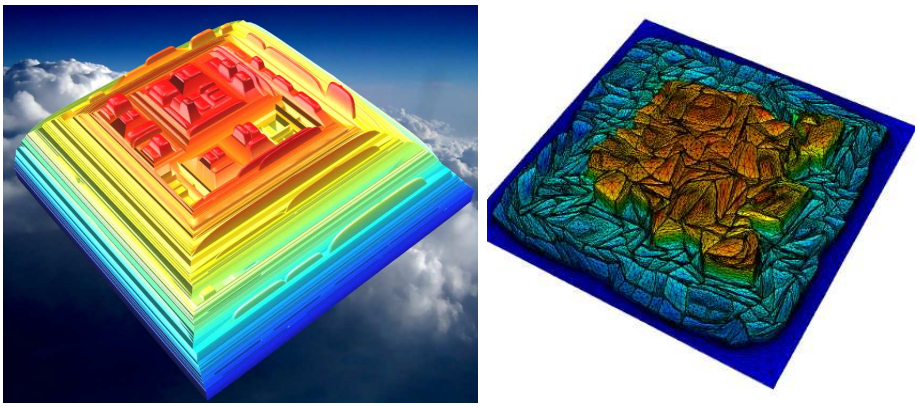
Going forward, we are focusing our attention on using the technique for visualization and analysis of high-dimensional data such as protein folding and biomedical applications. We plan on extending the algorithm to incorporate additional geometric information that might be obtained through dimensionality reduction techniques.



(a)



(b) (c)



(d) (e)

Figure 10: (a) Space of possible visualizations for the protein dataset, revealing how different choices of x_∞ can impact the discovery of certain features of the dataset such as the funnel structure. (b) Result of the proposed method that reveals the folding funnel. (c) An alternative view of the data that accentuates the folding funnel and the structures surrounding local maxima. (d) Result of the proposed algorithm which accentuates local maxima at the expense of revealing the folding funnel. (e) Result of [1] which maps the global minimum to the boundary of the terrain, and is directly comparable to (d).

8 Acknowledgements

This work is supported in part by the Department of Energy (DOE) under grant DE-FG02-06ER25735, by the National Science Foundation (NSF) under grants CCF-0747082 and DBI-0750891. We would like to thank Gunther Weber and Peer-Timo Bremer for donating their time and effort to produce terrain models of the protein datasets. We would also like to thank Rich Lehoucq, Kristi Maschhoff, Danny Sorensen, and Chao Yang for making available the ARPACK software [15] publicly, Josh Levine and Issam Safa for fruitful discussions, and Rephael Wenger for feedback and assistance with volumetric datasets. The background image used in the terrain models of this paper was provided by Arun Kulshreshtha under the Creative Commons Attribution 3.0 United States license (http://creativecommons.org/licenses/by/3.0/us/).

References

[1] Topological landscapes: A terrain metaphor for scientific data. *IEEE Transactions on Visualization and Computer Graphics*, 13(6):1416–1423, 2007. Member-Weber, Gunther and Member-Bremer, Peer-Timo and Member-Pascucci, Valerio.

[2] Chandrajit L. Bajaj, Valerio Pascucci, and Daniel R. Schikore. The contour spectrum. pages 167–173, 1997.

[3] Michael Balzer, Oliver Deussen, and Claus Lewerentz. Voronoi treemaps for the visualization of software metrics. In Thomas L. Naps and Wim De Pauw, editors, *SOFTVIS*, pages 165–172. ACM, 2005.

[4] Mikhail Belkin and Partha Niyogi. Laplacian eigenmaps for dimensionality reduction and data representation. *Neural Computation*, 15(6):1373–1396, 2003.

[5] Roger L. Boyell and Henry Ruston. Hybrid techniques for real-time radar simulation. In *AFIPS '63 (Fall): Proceedings of the November 12-14, 1963, fall joint computer conference*, pages 445–458, New York, NY, USA, 1963. ACM.

[6] Hamish Carr and Jack Snoeyink. Path seeds and flexible isosurfaces using topology for exploratory visualization. In *VISSYM '03: Proceedings of the symposium on Data visualisation 2003*, pages 49–58, Aire-la-Ville, Switzerland, Switzerland, 2003. Eurographics Association.

[7] Hamish Carr, Jack Snoeyink, and Ulrike Axen. Computing contour trees in all dimensions. *Comput. Geom*, 24(2):75–94, 2003.

[8] Hamish Carr, Jack Snoeyink, and Michiel van de Panne. Simplifying flexible isosurfaces using local geometric measures. In *IEEE Visualization*, pages 497–504. IEEE Computer Society, 2004.

[9] Scott E. Dillard. Topology-controlled volume rendering. *IEEE Transactions on Visualization and Computer Graphics*, 13(2):330–341, 2007. Member-Weber, Gunther H. and Member-Carr, Hamish and Member-Pascucci, Valerio and Member-Hamann, Bernd.

[10] Christopher Gold and Sean Cormack. Spatially ordered networks and topographic reconstruction. In Duane Marble, editor, *Proceedings of the Second International Symposium on Spatial Data Handling*, pages 74–85, Seattle, Washington, 1986.

[11] Teofilo F. Gonzalez. Clustering to minimize the maximum intercluster distance. *Theoretical Computer Science*, 38(2–3):293–306, June 1985.

[12] Leo Grady, Thomas Schiwietz, Shmuel Aharon, and Rüdiger Westermann. Random walks for interactive alpha-matting. In J. J. Villanueva, editor, *Proceedings of the Fifth IASTED International Conference on Visualization, Imaging and Image Processing*, pages 423–429, Benidorm, Spain, Sept. 2005. ACTA Press.

[13] Harold Hotelling. Analysis of a complex of statistical variables into principal components. *Journal of Educational Psychology*, 24:417–441, 498–520, 1933.

[14] B. Johnson and Ben Shneiderman. Tree maps: A space-filling approach to the visualization of hierarchical information structures. In *IEEE Visualization*, pages 284–291, 1991.

[15] R. B. Lehoucq, D. C. Sorensen, and C Yang. ARPACK USERS GUIDE: Solution of large scale eigenvalue problems by implicitly restarted arnoldi methods. Available at `http//www.caam.rice.edu/software/ARPACK/index.html`, 1997.

[16] Peter Lindstrom and Valerio Pascucci. Terrain simplification simplified: A general framework for view-dependent out-of-core visualization. *IEEE Trans. Vis. Comput. Graph*, 8(3):239–254, 2002.

[17] J. Milnor. *Morse Theory*, volume 51 of *Annals of mathematics studies*. Princeton University Press, Princeton, 1963.

[18] Krzysztof Onak and Anastasios Sidiropoulos. Circular partitions with applications to visualization and embeddings. In Monique Teillaud, editor, *Symposium on Computational Geometry*, pages 28–37. ACM, 2008.

[19] Valerio Pascucci. On the topology of the level sets of a scalar field. In *CCCG*, pages 141–144, 2001.

[20] S. T. Roweis and L. K. Saul. Nonlinear dimensionality reduction by locally linear embedding. *Science*, 290(5500):2323–2326, December 2000.

[21] Shigeo Takahashi, Issei Fujishiro, and Masato Okada. Applying manifold learning to plotting approximate contour trees. *IEEE Transactions on Visualization and Computer Graphics*, 15(6), 2009.

[22] Shigeo Takahashi, Tetsuya Ikeda, Yoshihisa Shinagawa, Tosiyasu L. Kunii, and Minoru Ueda. Algorithms for extracting correct critical points and constructing topological graphs from discrete geographical elevation data. *Comput. Graph. Forum*, 14(3):181–192, 1995.

[23] Tarasov and Vyalyi. Construction of contour trees in 3D in O(n log n) steps. In *COMPGEOM: Annual ACM Symposium on Computational Geometry*, 1998.

[24] J. B. Tenenbaum, V. de Silva, and J. C. Langford. A global geometric framework for nonlinear dimensionality reduction. *Science*, 290(5500):2319–2323, December 2000.

[25] David Turo and Walter-Alexander Jungmeister. Adapting treemaps to stock portfolio visualization. Technical Report CS-TR-2996, University of Maryland, College Park, November 1992.

[26] L. J. P. van der Maaten, E. O. Postma, and H. J. van den Herik. Dimensionality reduction: A comparative review. Technical report, MICC, Maastricht University, P.O. Box 616, 6200 MD Maastricht, Netherlands, February 2008.

[27] van Kreveld, van Oostrum, Bajaj, Pascucci, and Schikore. Contour trees and small seed sets for isosurface traversal. In *COMPGEOM: Annual ACM Symposium on Computational Geometry*, 1997.

[28] Wattenberg, Martin. Visualizing the stock market. In *Proceedings of ACM CHI 99 Conference on Human Factors in Computing Systems*, volume 2 of *Late-breaking results: seeing is understanding: new visualization techniques*, pages 188–189, 1999.